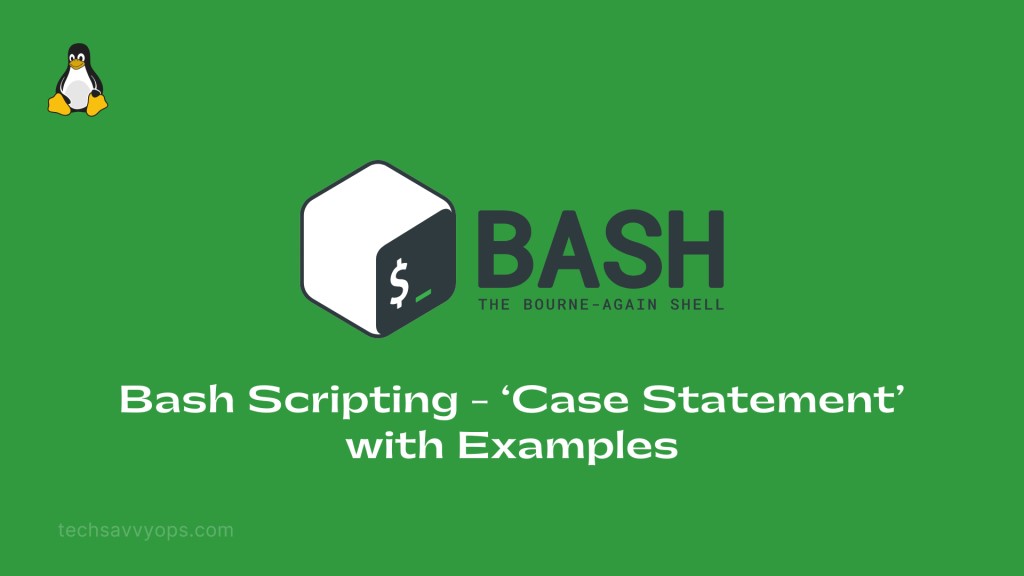
**Bash Scripting – Using ‘Case Statements’ with Example**

[**Pranav T**](https://techsavvyops.com/author/pranavtadakamalla/)



Bash scripting is an essential skill for any developer or system administrator, enabling the automation of repetitive tasks and simplifying complex workflows. One useful feature in Bash scripting is the **case statement**, which allows you to evaluate multiple conditions efficiently.

Let’s dive into a practical example that evaluates weekdays and outputs a message based on the input.

# The Scenario

In this script, users can input a day of the week (in capital letters), and the script will respond with a message indicating whether it’s a weekday or part of the weekend. Here’s how the script looks:

1 /bin/bash

2

3 echo "Enter a week day (capital letters)"

4

5 read day

6

1. case $day in
2. SUNDAY) echo "Weekend"
3. MONDAY) echo "ugh shit"
4. SATURDAY) echo "Weekend"
5. FRIDAY) echo "Almost weekend"
6. TUESDAY|WEDNESEDAY|THURSDAY) echo "Weekday"
7. \*) echo "incorrect entry"

14

15 esac

**Case Statement**: This structure simplifies handling multiple conditions by matching the value of a variable against several patterns.

**Pattern Matching**: Each pattern (e.g., SUNDAY, MONDAY) ends with ) and is followed by the corresponding action.

**Default Case (\*)**: This handles any input that doesn’t match predetermined patterns, ensuring robust error handling.

# How It Works

1. The script prompts the user to enter a day of the week in all capital letters.
2. The input is read into the variable day.
3. The case statement evaluates the value of the day:

For SUNDAY or SATURDAY, it prints Weekend.

For MONDAY, it humorously outputs ugh shit.

For FRIDAY, it acknowledges the proximity to the weekend with Almost Weekend.

For TUESDAY, WEDNESDAY, or THURSDAY, it identifies them as weekdays.

If the input doesn’t match any of these patterns, it falls back to the default case and outputs an incorrect entry.

# Running the Script

1. Save the script in a le, for example, weekday.sh.
2. Make the script executable by running:

chmod x weekday.sh

1. Execute the script:

./weekday.sh

# Enhancing the Script

You can further improve this script by:

1. Allowing case-insensitive input by converting input to uppercase:
   1. read day
   2. day=$(echo "$day" | tr '[:lower:]' '[:upper:]')
2. Adding more nuanced messages for specific days or events.
3. Expanding error handling to include suggestions for correct input.

# Final Thoughts

This simple Bash script demonstrates the power and flexibility of the case statement for handling multiple conditions. Whether you’re automating server tasks or just creating fun scripts for practice, mastering case will make your Bash scripting more efficient and readable. Try customizing this script to suit your needs and see how far you can take it!